PYTHON IMPLEMENTATION OF
WENO INTERPOLATION &
RECONSTRUCTION

Presented by: Adrian Townsend

In collaboration with:
Professor Randy LeVeque
David Ketcheson

University of Washington
Outline

WENO:
- Introduction
- The Big Picture
- Examples, Advantages & Costs

Implementation:
- Motivation for Python Approach
- A New Class: Piecewise Rational Class
- Present & Future Work
Weighted Essentially NonOscillatory Schemes

An adaptive interpolation or reconstruction procedure.

- Achieve arbitrarily high order of formal accuracy in smooth regions
- Maintain stable, nonoscillatory and sharp discontinuity transitions

Popular with solving hyperbolic conservation laws

\[
\frac{d}{dt} \bar{u}_i + \frac{1}{\Delta x} \left[ F(u_{i+1}) - F(u_i) \right] = 0
\]
WENO – The Big Picture

Given (or computed): \( u_i \) or \( \bar{u}_i \) (average of cell between \( x_i \) and \( x_{i+1} \))

Desired: Function that can be evaluated for any \( x \)

- Interpolate or Reconstruct (from cell averages)
  - Find rational interpolant, \( p(x) \)
  - Arbitrary order of accuracy, \( \mathcal{N}=5 \)

\[
\begin{align*}
\text{S} & \quad \bar{u}_i \quad \text{S} \\
S_1 & \quad x_i \quad x_{i+1} \\
S_2 & \quad \text{S}_3 \\
\end{align*}
\]

\[
\begin{align*}
\quad u(x_i) & \approx p(x_i) \\
\quad u(x_i) & \approx \sum_{k=1}^{n} \omega_k \, p_k(x_i)
\end{align*}
\]
Advantages & Costs

- Arbitrarily high order of accuracy
- Adaptive (nonlinear weights)
- Ideal for convection dominated problems with sharp discontinuities and complicated smooth solution structures.
- Computationally Expensive!
WENO versus scipy.interpolate.spline Interpolation

Black is f(x), blue is WENO, degree 3

Black is f(x), blue is WENO, degree 5
WENO Reconstruction from averages

dots are cell averages, blue is WENO, degree 3

dots are cell averages, blue is WENO, degree 5
Implementing WENO

Given order of accuracy, $\mathcal{N} = 2^n - 1$, $x_i$'s, $u_i$'s

- Find $n$ polynomial interpolants, $p_k(x_i)$, for $u(x_i)$ of $O(n)$ on each $S_k$
  - ($p_k$ degree $n-1$)

- Find nonlinear weights $\omega_k(x) = \frac{(\beta_k + \varepsilon)^{-1}\alpha_k(x)}{\sum_{k=1}^{n}(\beta_k + \varepsilon)^{-1}\alpha_k(x)}$
  - $\alpha_k = \alpha_k(x_i, \mathcal{N})$ is a polynomial
  - $\beta_k = \beta_k(p_k, x_i, x_{i+1})$ is a smoothness factor that involves integrating derivatives of $p_k$ over the interval $[x_i, x_{i+1}]$

A solution: return piecewise rational object

* Chi-Wang Shu, 2009
class PiecewiseRational(object):
    def __init__(self, xbreak=[], p_num=None, p_denom=None):
        ""
        INPUT:
        xbreak = list of break points, can be empty
        p_num = list of np.poly1d polynomials of length len(xbreak)+1
        p_denom = list of np.poly1d polynomials of length len(xbreak)+1
            None ==> all p_num and p_denom should be initialized to p(x)=1 everywhere.
        ""

    def __call__(self, x):
        ""
        Evaluate pw rational at point x (or array of points).
        ""

    def xbreak_eval(self):
        ""
        Evaluate at breakpoints self.xbreak and return left and right limits at each point
        in arrays yleft and yright.
        OUTPUT: (yleft, yright) tuple of lists.
        ""
def plot(self, xmin=None, xmax=None, npts=None, xmarginfactor=0.1):
    """
    Plot the piecewise rational function on the interval [xmin, xmax] using npts
    points on each interval between breakpoints.
    """

def weno_interp(xi, yi, degree=5, method='weno', Epsilon=1e-6):
    """
    Construct a piecewise polynomial function interpolating the values yi at points xi.
At Present & Looking Forward

- **Currently:**
  - Implementing boundary condition approaches
  - Comparing two approaches for reconstruction
  - Documentation

- **Later:**
  - Submit module to SciPy
Thank You

Contact:
Adrian Townsend  adrian.townsend@gmail.com
Prof. Randy LeVeque  rjl@amath.washington.edu
References


Stencil Growth

$k = 4$
Order $2k+1 = 9$
$k+1 = 5$ Stencils

$k = 3$
Order $2k+1 = 7$
$k+1 = 4$ Stencils

$k = 2$
Order $2k+1 = 5$
$k+1 = 3$ Stencils

$k = 1$
Order $2k+1 = 3$
$k+1 = 2$ Stencils